WHAT IS CLAIMED:

1. A method of generating an intermediate representation of program code, comprising the steps of:

decoding instructions in the program code;

generating an intermediate representation (IR) of the decoded program code to include at least one type of IR nodes out of a plurality of possible types of IR nodes; and

determining which type of IR nodes to generate in the intermediate representation for each respective instruction in the decoded program code,

wherein the IR nodes in the intermediate representation (IR) are abstract representations of the expressions, calculations, and operations performed by the program code.

2. The method of claim 1, wherein the plurality of possible types of IR nodes include base nodes and complex nodes.

3. The method of claim 2, wherein base nodes represent the most basic semantics of any subject architecture running the program code, such that the semantics of base nodes cannot be decomposed into other nodes representing more simple semantics.

4. The method of claim 3, wherein base nodes are generic across a plurality of possible subject architectures.

5. The method of claim 3, wherein complex nodes provide a more compact representation of the semantics of complex instructions in the program code than that of base node representations.

6. The method of claim 5, wherein complex nodes represent immediate type instructions in which a constant operand value is encoded into the immediate type instruction itself in an immediate field.

7. The method of claim 5, wherein a complex node may be decomposed into a plurality of base nodes to represent the same semantics of an instruction in the decoded program code.

8. The method of claim 5, wherein the program code is designed to be executed by a subject architecture, the method further comprising the step of generating complex nodes only for those features correspondingly configurable on the subject architecture.

9. The method of claim 2, wherein the plurality of possible types of IR nodes further include polymorphic nodes.

10. The method of claim 9, wherein the program code is subject code designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture, said method further comprising:

generating the intermediate representation to include polymorphic nodes,

wherein polymorphic nodes contain a function pointer to a function of the target architecture specific to a particular instruction in the subject code.

11. The method of claim 10, said method further comprising generating polymorphic nodes when the features of the target architecture would cause the semantics of a particular subject instruction to be lost if realized as base nodes.

12. The method of claim 10, wherein each polymorphic node is specific to a combination of a particular instruction in the subject code and a function of the target architecture.

13. The method of claim 10, wherein said determining the type of IR nodes step further comprises identifying an instruction in subject code which corresponds an instruction on a list of polymorphic instructions to be realized as polymorphic nodes; and

when a subject instruction corresponds to an instruction on the list of polymorphic instructions, said IR generating step generates polymorphic nodes only for those subject instructions corresponding to those on the list of polymorphic instructions.

14. The method of claim 1, wherein the plurality of possible types of IR nodes further include base nodes and architecture specific nodes.

15. The method of claim 14, wherein the program code is subject code designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture, said method further comprising:

generating the intermediate representation to include architecture specific nodes which are specific to a particular combination of a subject architecture and a target architecture.

16. The method of claim 15, the intermediate representation generating step further comprising:

initially representing all of the instructions in the subject code as subject architecture specific nodes, where each subject architecture specific node corresponds to a respective instruction in the subject code;

determining whether an instruction in the subject code is one in which to provide a target architecture specialized conversion function,

converting subject architecture specific nodes into target architecture specific nodes for those instructions determined to provide a target architecture specialized conversion function; and

generating base nodes from the remaining subject architecture specific nodes which are not identified as providing a target architecture specialized code generation function.

17. The method of claim 16, further comprising generating corresponding target code from the target architecture specific nodes which is specialized for the target architecture.

18. The method of claim 15, further comprising generating corresponding target code from the base nodes which is not specialized for the target architecture.

19. A computer readable recording medium containing program code for performing the method of claim 1.

20. A computer readable storage medium having translator software resident thereon in the form of computer readable code executable by a computer to perform the following steps during translation of subject program code to target program code:

decoding instructions in the subject program code;

generating an intermediate representation (IR) of the decoded subject program code to include at least one type of IR nodes out of a plurality of possible types of IR nodes;

determining which type of IR nodes to generate in the intermediate representation for each respective instruction in the decoded subject program code,

wherein the IR nodes in the intermediate representation (IR) are abstract representations of the expressions, calculations, and operations performed by the program code; and

generating target program code using the intermediate representation (IR).

21. The computer readable storage medium of claim 20, wherein the plurality of possible types of IR nodes include base nodes and complex nodes.

22. The computer readable storage medium of claim 21, wherein base nodes represent the most basic semantics of any subject architecture running the program code, such that the semantics of base nodes cannot be decomposed into other nodes representing more simple semantics.

23. The computer readable storage medium of claim 22, wherein base nodes are generic across a plurality of possible subject architectures.

24. The computer readable storage medium of claim 22, wherein complex nodes provide a more compact representation of the semantics of complex instructions in the program code than that of base node representations.

25. The computer readable storage medium of claim 24, wherein complex nodes represent immediate type instructions in which a constant operand value is encoded into the immediate type instruction itself in an immediate field.

26. The computer readable storage medium of claim 24, wherein a complex node may be decomposed into a plurality of base nodes to represent the same semantics of an instruction in the decoded program code.

27. The computer readable storage medium of claim 24, wherein the subject program code is designed to be executed by a subject architecture, the method further

comprising the step of generating complex nodes only for those features correspondingly configurable on the subject architecture.

28. The computer readable storage medium of claim 21, wherein the, plurality of possible types of IR nodes further include polymorphic nodes.

29. The computer readable storage medium of claim 28, wherein the subject program code is designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture, said translator software further containing computer readable code executable by a computer to' perform the following steps:

generating the intermediate representation to include polymorphic nodes,

wherein polymorphic nodes contain a function pointer to a function of the target architecture specific to a particular instruction in the subject code.

30. The computer readable storage medium of claim 29, said translator software further containing computer readable code executable by a computer to generate polymorphic nodes when the features of the target architecture would cause the semantics of a particular subject instruction to be lost if realized as base nodes.

31. The computer readable storage medium of claim 29, wherein each polymorphic node is specific to a combination of a particular instruction in the subject code and a function of the target architecture.

32. The computer readable storage medium of claim 29, wherein said computer readable code executable by a computer for determining the type of IR nodes further:

identifies an instruction in subject code which corresponds an instruction on a list of polymorphic instructions to be realized as polymorphic nodes; and

when a subject instruction corresponds to an instruction on the list of polymorphic instructions, generates polymorphic nodes only for those subject instructions corresponding to those on the list of polymorphic instructions.

33. The computer readable storage medium of claim 20, wherein the plurality of possible types of IR nodes further include base nodes and architecture specific nodes.

34. The computer readable storage medium of claim 33, wherein the subject program code is designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture, said translator software further containing computer readable code executable by a computer to perform the following steps:

generating the intermediate representation to include architecture specific nodes which are specific to a particular combination of a subject architecture and a target architecture.

35. The computer readable storage medium of claim 34, said translator software further containing computer readable code executable by a computer to perform the following steps:

initially representing all of the instructions in the subject code as subject architecture-specific nodes, where each subject architecture specific node corresponds to a respective instruction in the subject code;

determining whether an instruction in the subject code is one in which to provide a target architecture specialized conversion function,

converting subject architecture specific nodes into target architecture specific nodes for those instructions determined to provide a target architecture specialized conversion function; and

generating base nodes from the remaining subject architecture specific nodes which are not identified as providing a target architecture specialized code generation function.

36. The computer readable storage medium of claim 35, said translator software further containing computer readable code executable by a computer to generate corresponding target code from the target architecture specific nodes which is specialized for the target architecture.

37. The computer readable storage medium of claim 34, said translator software further containing computer readable code executable by a computer to generate corresponding target code from the base nodes which is not specialized for the target architecture.

38. A translator apparatus for use in a target computing environment having a processor and a memory coupled to the processor for translating subject program code appropriate in a subject computing environment to produce target program code appropriate to the target computing environment, the translator apparatus comprising:

a decoding mechanism configured to decode instructions in the subject program code;

an intermediate representation generating mechanism configured to generate an intermediate representation (IR) of the decoded program code to include at least one type of IR nodes out of a plurality of possible types of IR nodes; and

an intermediate representation (IR) type determining mechanism configured to determine which type of IR nodes to generate in the intermediate representation for each respective instruction in the decoded program code,

wherein the IR nodes in the intermediate representation (IR) are abstract representations of the expressions, calculations, and operations performed by the program code.

39. The translator apparatus of claim 38, wherein the plurality of possible types of IR nodes include base nodes and complex nodes.

40. The translator apparatus of claim 39, wherein base nodes represent the most basic semantics of any subject architecture running the program code, such that the semantics of base nodes cannot be decomposed into other nodes representing more simple semantics.

41. The translator apparatus of claim 40, wherein base nodes are generic across a plurality of possible subject architectures.

42. The translator apparatus of claim 40, wherein complex nodes provide a more compact representation of the semantics of complex instructions in the program code than that of base node representations.

43. The translator apparatus of claim 42, wherein complex nodes represent immediate type instructions in which a constant operand value is encoded into the immediate type instruction itself in an immediate field.

44. The translator apparatus of claim 42, wherein a complex node may be decomposed into a plurality of base nodes to represent the same semantics of an instruction in the decoded program code.

45. The translator apparatus of claim 42, wherein the program code is designed to be executed by a subject architecture, the intermediate representation generating mechanism further comprising a complex node generating mechanism for

generating complex nodes only for those features correspondingly configurable on the subject architecture.

46. The translator apparatus of claim 39, wherein the plurality of possible types of IR nodes further include polymorphic nodes.

47. The translator apparatus of claim 46, wherein the program code is subject code designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture, the intermediate representation generating mechanism further comprising:

a polymorphic node generating mechanism for generating the intermediate representation to include polymorphic nodes,

wherein polymorphic nodes contain a function pointer to a function of the target architecture specific to a particular instruction in the subject code.

48. The translator apparatus of claim 47, said polymorphic node generating mechanism generating polymorphic nodes when the features of the target architecture would cause the semantics of a particular subject instruction to be lost if realized as base nodes.

49. The translator apparatus of claim 47, wherein each polymorphic node is specific to a combination of a particular instruction in the subject code and a function of the target architecture.

50. The translator apparatus of claim 47, wherein said intermediate representation (IR) type determining mechanism further comprises a polymorphic identification mechanism for identifying an instruction in subject code which corresponds an instruction on a list of polymorphic instructions to be realized as polymorphic nodes; and

when a subject instruction corresponds to an instruction on the list of polymorphic instructions, said intermediate representation generating mechanism generates polymorphic nodes only for those subject instructions corresponding to those on the list of polymorphic instructions.

51. The translator apparatus of claim 38, wherein the plurality of possible types of IR nodes further include base nodes and architecture specific nodes.

52. The translator apparatus of claim 51, wherein the program code is subject code designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture, said intermediate representation generating mechanism further comprising:

an architecture specific node generating mechanism for generating the intermediate representation to include architecture specific nodes which are specific to a particular combination of a subject architecture and a target architecture.

53. The translator apparatus of claim 52, the intermediate representation generating mechanism being configured to:

initially represent all of the instructions in the subject code as subject architecture-specific nodes, where each subject architecture specific node corresponds to a respective instruction in the subject code;

determine whether an instruction in the subject code is one in which to provide a target architecture specialized conversion function,

convert subject architecture specific nodes into target architecture specific nodes for those instructions determined to provide a target architecture specialized conversion function; and

generate base nodes from the remaining subject architecture specific nodes which are not identified as providing a target architecture specialized code generation function.

54. The translator apparatus of claim 53, further comprising a specialized target code generating mechanism for generating corresponding target code from the target architecture specific nodes which is specialized for the target architecture.

55. The translator apparatus of claim 52, further comprising a non specialized target code generating mechanism for generating corresponding target code from the base nodes which is not specialized for the target architecture.

56. The translator apparatus of claim 47, wherein said generated polymorphic nodes specify the registers to be allocated during target code generation.

57. The translator apparatus of claim 47, wherein said generated polymorphic nodes are utilized in generic kernel optimizations by inferring information from the function pointer in the polymorphic node which may otherwise be indeterminable from the polymorphic node.

58. The translator apparatus of claim 50, wherein when a subject instruction corresponds to an instruction on the list of polymorphic instructions, said intermediate representation generating mechanism generates either polymorphic nodes or base nodes for those subject instructions corresponding to those on the list of polymorphic instructions.

59. The method of claim 10, wherein said generated polymorphic nodes specify the registers to be allocated during target code generation.

60. The method of claim 10, wherein said generated polymorphic nodes are utilized in generic kernel optimizations by inferring information from the function pointer

in the polymorphic node which may otherwise be indeterminable from the polymorphic node.

61. The method of claim 13, wherein when a subject instruction corresponds to an instruction on the list of polymorphic instructions, said intermediate representation generating step generates either polymorphic nodes or base nodes for those subject instructions corresponding to those on the list of polymorphic instructions.

62. The computer-readable storage medium of claim 29, wherein said generated polymorphic nodes specify the registers to be allocated during target code generation.

63. The computer-readable storage medium of claim 29, wherein said generated polymorphic nodes are utilized in generic kernel optimizations by inferring information from the function pointer in the polymorphic node which may otherwise be indeterminable from the polymorphic node.

64. The translator apparatus of claim 32, wherein said computer readable code executable by a computer for determining the type of IR nodes further, when a subject instruction corresponds to an instruction on the list of polymorphic instructions, generates either polymorphic nodes or base nodes for those subject instructions corresponding to those on the list of polymorphic instructions.

65. A method of translating subject program code capable of being executed on a subject processor architecture to target program code capable of being executed on a target processing architecture using a translator configurable between a plurality of possible subject/target processing architecture pairings, said method comprising:

selecting a subject processor architecture on which the subject program code is designed to be executed from a plurality of possible subject processor architectures;

selecting a target processor architecture on which the target program code is to be executed from a plurality of possible target processor architectures; and

configuring a translator to translate the subject program code to target program code using a pairing of the selected subject processor architecture and the selected target processor architecture.

66. The method of claim 65, further comprising translating the subject program code to target program code dynamically at run-time while the target program code is being executed on the target processing architecture.

67. The method of claim 65, further comprising:

decoding instructions in the subject program code;

determining which types of intermediate representation (IR) nodes out of a plurality of possible types of IR nodes to utilize in an intermediate representation of the decoded program code for each respective instruction in the decoded program code based upon the particular translator configuration being undertaken based on the pairing of the selected subject processor architecture and the selected target processor architecture; and

generating an intermediate representation (IR) of the decoded program code to include at least one type of IR nodes out of a plurality of possible types of IR nodes;

wherein the IR nodes in the intermediate representation (IR) are abstract representations of the expressions, calculations, and operations performed by the program code.

68. The method of claim 67, further comprising generating the intermediate representation (IR) to include a combination of generic conversion features and specific conversion features, wherein generic conversion features are capable of being implemented across a plurality of possible processor architectures while specific conversion features are capable of being implemented by a specific processor architecture.

69. The method of claim 68, wherein the particular translator configuration being undertaken determines the respective combination of generic conversion features and specific conversion features utilized.

70. A computer readable storage medium having translator software resident thereon in the form of computer readable code executable by a computer for performing a method of translating subject program code capable of being executed on a subject processor architecture to target program code capable of being executed on a target processing architecture using a translator configurable between a plurality of possible subject/target processing architecture pairings, said method comprising:

selecting a subject processor architecture on which the subject program code was designed to be executed from a plurality of possible subject processor architectures;

selecting a target processor architecture on which the target program code is to be executed from a plurality of possible target processor architectures; and

configuring a translator to translate the subject program code to target program code using a pairing of the selected subject processor architecture and the selected target processor architecture.

71. The computer-readable storage medium of claim 70, said translator software further containing computer readable code executable by a computer to translate the subject program code to target program code dynamically at run-time while the target program code is being executed on the target processing architecture.

72. The computer-readable storage medium of claim 70, said translator software further containing computer readable code executable by a computer to perform the following steps:

decoding instructions in the subject program code;

determining which types of intermediate representation (IR) nodes out of a plurality of possible types of IR nodes to utilize in an intermediate representation of the decoded program code for each respective instruction in the decoded program code based upon the particular translator configuration being undertaken based on the pairing of the selected subject processor architecture and the selected target processor architecture; and

generating an intermediate representation (IR) of the decoded program code to include at least one type of IR nodes out of a plurality of possible types of IR nodes;

wherein the IR nodes in the intermediate representation (IR) are abstract representations of the expressions, calculations, and operations performed by the program code.

73. The computer-readable storage medium of claim 72, said translator software further containing computer readable code executable by a computer to generate the intermediate representation (IR) to include a combination of generic conversion features and specific conversion features, wherein generic conversion features are capable of being implemented across a plurality of possible processor architectures while specific conversion features are capable of being implemented by a specific processor architecture.

74. The computer-readable storage medium of claim 73, wherein the particular translator configuration being undertaken determines the respective combination of generic conversion features and specific conversion features utilized.

75. A translator apparatus for use in a target computing environment having a processor and a memory coupled to the processor for translating subject program code capable of being executed on a subject processor architecture to target program code capable of being executed on the target processor architecture of the target computing environment using a translator configurable between a plurality of possible subject/target processing architecture pairings, the translator apparatus comprising:

a subject processor selecting mechanism configured to select a subject processor architecture on which the subject program code was designed to be executed from a plurality of possible subject processor architectures;

a target processor selecting mechanism configured to select a target processor architecture on which the target program code is to be executed from a plurality of possible target processor architectures; and

a configuration mechanism configured to configure a translator to translate the subject program code to target program code using a pairing of the selected subject processor architecture and the selected target processor architecture.

76. The translator apparatus of claim 75, further comprising a translation mechanism configured to translate the subject program code to target program code dynamically at run-time while the target program code is being executed on the target processing architecture.

77. The translator apparatus of claim 75, further comprising:

a decoding mechanism configured to decode instructions in the subject program code;

an intermediate representation (IR) type determining mechanism configured to determine which types of intermediate representation (IR) nodes out of a plurality of possible types of IR nodes to utilize in an intermediate representation of the decoded program code for each respective instruction in the decoded program code based upon the particular translator configuration being undertaken based on the pairing of the selected subject processor architecture and the selected target processor architecture; and

an intermediate representation (IR) generating mechanism configured to generate an intermediate representation (IR) of the decoded program code to include at least one type of IR nodes out of a plurality of possible types of IR nodes;

wherein the IR nodes in the intermediate representation (IR) are abstract representations of the expressions, calculations, and operations performed by the program code.

78. The translator apparatus of claim 77, wherein the intermediate representation (IR) generating mechanism is further configured to generate the intermediate representation (IR) to include a combination of generic conversion features and specific conversion features, wherein generic conversion features are capable of being implemented across a plurality of possible processor architectures while specific conversion features are capable of being implemented by a specific processor architecture.

79. The translator apparatus of claim 78, wherein the particular translator configuration being undertaken determines the respective combination of generic conversion features and specific conversion features utilized.